PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classificati n 7 : A63F 13/06, G06F 3/033

A1

(11) International Publication Number: WO 00/59594

(43) International Publication Date: 12 October 2000 (12.10.00)

(21) International Application Number: PCT/US00/08848

(22) International Filing Date: 3 April 2000 (03.04.00)

(30) Priority Data:

60/127,972	6 April 1999 (06.04.99)	US
09/483,113	14 January 2000 (14.01.00)	US
09/497,914	4 February 2000 (04.02.00)	US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052 (US).

(72) Inventors: ANDREWS, Marcus, J.; 9206 162nd Place NE, Redmond, WA 98052 (US). FIRDOSH, Bhesania, K.; Apartment H-202, 10820 113th Court NE, Kirkland, WA 98033 (US). HOLAN, Doron, J.; 1266 NE 87th, Kirkland, WA 98033 (US). INGMAN, Robert; 1121 37th Avenue, Seattle, WA 98122 (US). LEATHAM, Scott, R.; 13029 326th Avenue NE, Duvall, WA 98019 (US). PERETZ, Ervin; 3649 E. Ames Lake Lane, Redmond, WA 98053 (US). RAY, Kenneth, D.; 8127 149th Place NE #D320, Redmond, WA 98052 (US). SHARMA, Om, K.; 10933 127th Place NE, Kirkland, WA 98033 (US). VERES, James, E.; 14029 171st Lane NE, Woodinville, WA 98072 (US).

(74) Agents: HAYES, Daniel, L. et al.; Suite 500, 421 W. Riverside Avenue, Spokane, WA 99201 (US).

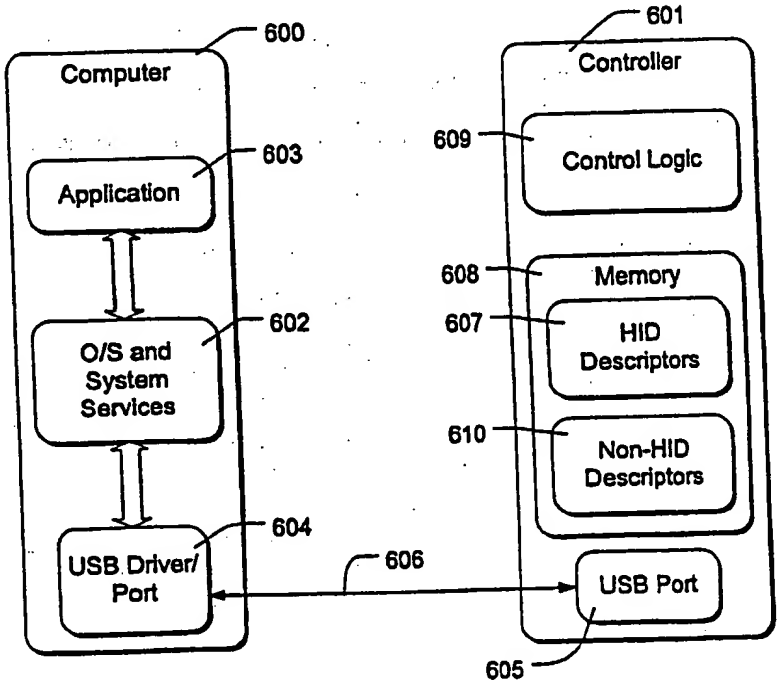
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: GAME CONTROL DEVICE HAVING GENRE DATA



(57) Abstract

A computer peripheral has a processor, non-volatile memory, and a plurality of controls. The non-volatile memory holds control mappings corresponding to a plurality of application program genres. The control mappings indicate actions to be performed in application

GAME CONTROL DEVICE HAVING GENRE DATA

RELATED APPLICATIONS

This is a continuation-in-part application of a prior US Patent Application filed January 10, 2000, titled "Mapping Input-Device Functions to Software Application Input Commands," serial number 09/483,113. Priority is hereby claimed to this earlier application.

TECHNICAL FIELD

This invention relates generally to the use of peripheral input devices with software applications and more specifically to the use of genres in conjunction with such input devices and software applications.

BACKGROUND OF THE INVENTION

Various input devices are available that permit a computer user to communicate with a computer. A typical personal computer offers input devices such as a keyboard and a mouse. Numerous other devices are available, such as drawing pads, joysticks, and steering wheels (for use with driving games). These devices can be connected to a computer, and they permit the user to communicate information to the computer; the information communicated instructs software applications running on the computer to perform specified actions.

Ideally, a computer user loads a software application, connects an appropriate device to the computer, and the device and software work together without further configuration by the user. This ideal, however, has not been realized in prior systems.

applications on the system and configures the device to work better with those applications. However, these ad hoc approaches are error prone, may result in a relationship between device controls and software actions that feels unnatural to the user, and can only provide support for applications the device manufacturer knows about and chooses to support.

The Human Interface Device (HID) standard, defined in conjunction with the Universal Serial Bus (USB) standard, addresses problems such as these. Using HID, an input device can store information about its controls. This information can be obtained by a requesting application program.

HID data generally describes the characteristics of device controls and the format of data generated by the controls. In addition, HID allows different labels or "usages" to be assigned to the controls of an input device. For example, a button might be designated as a "fire" button, as "button1", "button2," etc. In addition, HID can include "aliases" for a control. A given control might have a plurality of aliases such as "button1," "fire," "torpedo," etc.

Usages and aliases, if used in a uniform and agreed upon way, provide useful hints about how to use a certain control on an input device. In some cases, for instance, the "fire" button will have an obviously corresponding action in an application program. In other cases, however, the usages and aliases might not correspond directly to any terminology known by the application program.

A further complicating factor is that there is no agreed upon standard for naming HID controls. Even if there were such a standard, it would remain—in may cases—a very difficult task for an application program to determine an optimum set of mappings between numerous input device controls and the actions available in the application program. A determination such as this would require a complex algorithm, tailored specifically for the application program.

the HID data. The actions specified in the non-HID data use these same string indexes to specify controls.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram representing a computer system in which aspects of the invention may be incorporated;

Fig. 2 is a block diagram showing the use of an input device mapper with input devices and software applications;

Fig. 3 is a block diagram showing a sample control-semantic correlation and its structure;

Fig. 4 is a block diagram showing a sample action-semantic correlation and its structure;

Fig. 5 is a block diagram showing a sample mapping created by an input device mapper;

Fig. 6 is an image of an input device with action labels;

Fig. 7 is a flowchart illustrating a process by which an input device mapper is used;

Fig. 8 is a block diagram showing the use of a mapping created by an input device mapping;

Fig. 9 is a block diagram of an exemplary computer and game control device;

Fig. 10 is a block diagram of a non-HID data descriptor;

Fig. 11 is a flowchart showing methodological aspects of the described embodiment.

Figs. 12 and 13 are flowcharts showing methodological aspects of a USB implementation.

basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 may further include a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD-ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, program data 38, and an input device mapper 39. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40, a pointing device 42, a drawing pad 65, or a game controller such as driving game controller 66. Other input devices (not shown) may include a microphone, joystick, game pad,

and other means of establishing a communications link between the computers may be used.

Input Device Mapper

Figure 2 depicts the use of an input device manager. Input device mapper 39 is a software module that provides an interface between application programs 36 and input devices, such as devices 42, 65, 66, and 67. Input device mapper 39 could be a system services component of an operating system running on computer 20, such as operating system 35, or a stand-alone software module, as shown in Fig. 2.

Input device mapper 39 is associated with one or more genre descriptions, such as genre descriptions 211-213. An application program genre is a collection of games having similarities in operation and input device usage. A genre description for a specific genre defines mappings between input device controls and actions to be performed in a game of the genre. These actions are defined in terms semantics or labels.

In the system described below, agreed upon genre semantics or labels are preferably publicized so that input device manufacturers and software developers can use input device mapper 39 in the manner described below to allow devices and software to work together. Specifically, a plurality of genres are defined and a set of possible actions is defined for each genre. This will be explained in more detail below.

In Fig. 2, genre description 211 corresponds to a "driving game" genre (corresponding to example genre 1 in the Examples section below). Genre description 212 corresponds to a "role playing" genre (corresponding to example genre 3 in the Examples section below). Genre description 213 corresponds to a

semantics selected from the defined semantics of the driving simulation genre. Architectural design application 36b provides an A-S correlation between its actions and the defined semantics of the CAD genre. In addition to A-S correlation 231, driving game application 36a also provides A-S correlation 232 between its actions and the defined semantics of the role-playing genre. Providing two different A-S correlations for a single application is appropriate when the application has two different phases that require different usage of the controls. For example, in driving game application 36a, the character begins by driving a car; this phase of the game is in driving simulation genre 211. Later, the character gets out of the car and explores on foot; this phase is in the role-playing genre 212.

Input device mapper 39 receives C-S correlations 221-225 and A-S correlations 231-233. Input device mapper 39 creates a mapping for each application program 36a, 36b, on computer 20. For example, in order to create mapping 220 for driving game application 36a, input device mapper 39 first selects an appropriate device for the driving game genre, by determining which devices have a C-S correlation for the driving simulation genre. If there is more than one device having a C-S correlation for driving simulation genre 211, such as driving game controller 66 and joystick 67, then input device mapper 39 selects one of these devices. The selection may be made in various ways, for example by selecting the first appropriate connected device that input device mapper 39 locates, or by consulting a database of preferred devices for each genre. For example, input device mapper 39 selects game controller 66 because it is the first device that it locates which supports driving simulation genre 211. Once the device is selected, input device mapper 39 uses C-S correlation 221 and A-S correlation 231 to map controls on game controller 66 into actions that driving game application 36a performs. Input device mapper 39 may create the mapping by performing a simple

described as "turn left or right" is associated with the semantic "STEER", "speed up" is associated with the semantic "ACCELERATE", etc.

Figure 5 depicts a sample mapping 220 created by input device mapper 39, which links the controls on game controller 66 with actions performed by driving game application 36a. Controls 301 are correlated with semantics, as defined in C-S correlation 221. The semantics are correlated with software actions 401, as defined in A-S correlation 231. In the example, "trigger 1" 502 on game controller 66 is correlated with the semantic "FIRE" 503, which, in turn, is correlated with the software action "fire machine guns" 504.

The detail of an entry in the mapping is shown in items 511-513. Each entry contains a controller code 511, an application code 512, and a label 513. The controller code 511 is the data that an input device generates when a particular control has been operated. For example, the game controller could signify that trigger 1 has been pressed by generating the number "1002." The application code 512 is the item of input that an application expects to receive as an instruction to perform a particular action. For example, the input "64379" could instruct driving game application 36a to fire machine guns. Label 513 is a text string provided by application program 36a, which is a plain language description of the action that application program 36a will perform upon receiving application code 512 as its input. For example, "fire machine guns" is a label describing the action that will be performed by driving game application 36a when trigger 1 is depressed. The labels are helpful for displaying a graphic representation of the mapping, as described below in the text accompanying Fig. 6.

A mapping can be bi-directional. For example, it will be observed that mapping 220 shows a software action "feedback" 509 mapped into the "vibrate" control 511 on game controller 66 through the semantic "RUMBLE" 510. A genre,

with the "DASHBOARD" semantic, driving game 36a may still correlate its "change dash display" action with the semantic "DASHBOARD," and input device mapper 39 will locate an appropriate auxiliary input for that action. In mapping 220, auxiliary input 501 is selected by input device mapper 39 to implement the "DASHBOARD" semantic. Auxiliary input 501 may be a key on keyboard 40, an unused control on game controller 66 such as control 505, a pop-up menu that the user can control with pointing device 42, or any other mechanism by which the user can communicate with computer 20.

The genres may be defined such that some semantics must be mapped to the primary input device selected by input device mapper 39 and may not be mapped to an auxiliary input outside of that device. For example, in the genres provided below in the Examples section, controls are divided into the categories "priority 1" and "priority 2." A priority 1 control is a control that must be implemented on the primary input device and may not be implemented by an auxiliary input. A priority 2 control is a control that may be implemented on the primary input device, if a control is available. For example, in the genre "driving sim without weapons" shown in below in the Examples section, steering is a priority 1 control, so the "steer" semantic must be implemented on the primary input device selected by input device mapper 39, such as game controller 66. However, "dashboard" is a priority 2 control, so it may be implemented by any type of auxiliary input. Some other controls, which may be designated as "priority 3," are never implemented by the device used for the mapping, and therefore the genres do not define semantics to correlate with these controls. For example, a game application may provide a pop-up menu to change the resolution mode of the screen (640x480, 1024x768, etc), select the background music accompanying the game, select weapons to be carried,

the user's choice as a general preference that joystick 67 should work similarly with all games in first-person genres (i.e., that button 602 should enable a cloaking device in any first-person game that offers a cloaking device). The user's preferences may be stored in a file or database for future use by the user. Additionally, storing the preferences in a file or database permits the preferences to be easily ported from computer 20 to any other machine on which input device mapper 39 has been implemented, thus permitting consistent mappings across several machines.

Figure 7 is a flowchart showing an example use of an input device mapper, and the steps to initiate its use. As shown in Fig. 6 and described in detail below, a device and an application both undergo a setup phase, in which they pass their respective C-S and A-S correlations to an input device mapper; the application program then receives and processes input in accordance with the mapping.

Steps 701 through 704 relate to the setup of an application program for use with input device mapper 39. An application program, such as driving game application 36a, begins execution at step 701. At step 702, the application creates an array correlating actions with semantics. For example, application 36a could create an array representing A-S correlation 231. The array created at step 702 is passed to input device mapper 39 at step 703.

One method of representing A-S correlation 231 in the array created as step 702 is to assign a unique value to each action and to each semantic. For example, the semantics in genre 211, which are used in A-S correlation 231 and C-S correlation 221, may be assigned unique values as follows: 1 represents "STEER", 2 represents "ACCELERATE", etc. In a programming environment that supports symbolic constants, such as C++, it is convenient to represent the values as symbols. Input device mapper 39 may define the set of available genres and assign

along with game controller 66 on a medium such as magnetic disk 29 or optical disk 31; this array can then be passed to input device mapper 39 at step 706 by loading it into computer 20 through magnetic drive 28 or optical drive 30. Alternatively, game controller 66 may be known to the designer of input device mapper 39, in which case the array may be built into input device mapper 39.

Step 704 takes place after steps 703 and 706 have been completed. After input device mapper 39 has received the arrays created at step 702 and the array created at step 705, it creates a mapping, such as mapping 220, by the process described above in the text accompanying Fig. 5. After the mapping has been created, input device mapper 39 may provide mapping information to application program 36a at step 704. The mapping information provided may include information about which device control is correlated with each application-defined action. If the mapping is provided to application program 36a at step 704, then application program 36a can use the mapping to convert notifications of device events into actions that application program36a performs. Alternatively, instead of providing the mapping to application program 36a, the mapping may be provided to an input device manager, which is depicted in Fig. 8 and described below, which uses the mapping to translate device event notifications into input for application program 36a. In the case where an input device manager is used to translate device event notifications into application program input, it is not necessary to provide the mapping to application program 36a, in which case step 704 can be omitted.

Following step 704, application program 36a begins its input loop 709, which comprises listening for input at step 707, processing the input at step 708, and returning to step 707 to listen for more input. When the mapping has been provided to application program 36a at step 704, application program 36a can use the mapping to process the input. In this case, application program would receive

position –32 at time = T2, etc.), or a parameter further describing the device event (e.g., in addition to data signifying that motion along the x-axis has occurred, input device manager 801 may also provide data indicating the magnitude and direction of the motion, or data indicating the resulting position of the control). An application, such as driving game application 36a, may be interested in this information. For example, the firing of a weapon may become more rapid after trigger 1 has been depressed for more than one second. A different game application might cause a pinball cue or a slingshot to be pulled back further the longer a button has been depressed.

Input device manager 801 may receive event notifications from multiple devices, and use the data received from multiple devices to report unified instructions to an application. By doing so, it allows an application to be controlled by various devices while allowing the application to view its input as if it came from a single "unified" device. For example, the auxiliary input used to implement the "change dash display" action correlated with the "DASHBOARD" semantic in driving game 36a could be the "D" key on keyboard 40 (not shown in Fig. 8). Input device manager 801 will receive notification that the "DASHBOARD" semantic has been pressed, and will translate this notification into an instruction to driving game application 36a. The source of the input is transparent to application 36a, which knows only that it has received the instruction to perform the action correlated with the semantic "DASHBOARD." Another possible application of reporting unified instructions from multiple devices is aggregation of devices (e.g., using a joystick 67 with a set of pedals that are physically connected to driving game controller 66). Alternatively, in some circumstances it may be useful for input device manager 801 not to unify the input from multiple devices and to inform application 36a which input device a particular input originated from, for example

EXAMPLES

The following are examples of genres. The semantics in each genre are divided into "priority 1" semantics and "priority 2" semantics, which are described below:

Genre 1: Combat Driving sim, with weapons

Priority 1 Controls

Semantic	Description
Steer	left / right
Accelerate	faster / slower
Brake	Brake
Weapons	select next weapon
Fire	fires selected weapon
Target	selects next available target
Menu	pause, show menu of priority 2 & 3 controls

Priority 2 Controls

Semantic	Description
Look	forward / right / backward / left
View	cycle through view options
Device	show device and controls
Dashboard	select next dashboard / heads-up display option
Press to talk	for voice communication
Up shift	select next higher gear
Down shift	select next lower gear Reverse from neutral

Genre 3: Role Playing

Priority 1 Controls

Semantic	Description
Move	forward / back / left / right
Get	pick up and carry item
Select Inventory	select next inventory item
Apply	use selected inventory item
Attack	
Cast	cast spell
Talk	communicate
Crouch	crouch, climb down, swim down
Jump	jump, climb up, swim up
Menu	pause, show menu of priority 2 & 3 controls

Priority 2 Controls

Semantic	Description
Look	forward or up / back or down / left / right (usually maps to point of view ("POV") on devices that have one)
Map	cycle through map options
Display	shows next on-screen display options, maps, etc.
Press to talk	voice communication (multi-player)
Rotate	turn body left / right
Device	displays input device and controls

Genre 5: Real Time Strategy

Priority 1 Controls

Semantic	Description
Scroll	up / down / left / right
Select	Select unit / object / item
Instruct	cycle through instructions
Apply	apply selected instruction
Team	select next team (cycle through all)
Building	select next building
Menu	pause, show menu of priority 2 & 3 controls

Priority 2 Controls

Semantic	Description
Map	cycle through map options
Display	shows next on-screen display options, maps, etc.
Press to talk	voice communication (multi-player)
Device	displays input device and controls

Priority 1 Controls - pitching

Semantic	Description
Aim	aim where to pitch
Select	cycle through pitch selections
Pitch In	throw pitch into strike zone
Pitch Out	throw pitch outside of strike zone
Base	select base to throw to
Throw	throw to base
Catch	catch hit ball
Menu	pause, show menu of priority 2 & 3 controls

Priority 1 Controls - fielding

Semantic	Description
Aim	aim where to run or throw
Nearest	switch to fielder nearest to the ball
Conservative Throw	make conservative throw
Aggressive Throw	make aggressive throw
Burst	invoke burst of speed
Jump	jump to catch ball
Dive	dive to catch ball
Menu	pause, show menu of priority 2 & 3 controls

Priority 2 Controls - fielding

(none)

Genre 8: 2D Object Control (CAD)

Priority 1 Controls

Semantic	Description
Move	move object or scroll view up / down / left / right
View	select between move and scroll
Zoom	in / out
Select	
Special 1	do first special operation
Special 2	do second special operation
Special 3	do third special operation
Special 4	do fourth special operation
Menu	pause, show menu of priority 2 & 3 controls

Priority 2 Controls

Semantic	Description
Rotate Z	rotate object or view clockwise / counterclockwise
Display	shows next on-screen display options, etc.
Device	displays input device and controls

HID Implementation

Fig. 9 shows an implementation of a computer and/or game system that is compatible with the Universal Serial Bus (USB) and Human Device Interface (HID) specifications. Both of these specifications are available from USB Implementers Forum, which has current administrative headquarters in Portland, Oregon (current Internet URL: www.usb.org).

The system of Fig. 9 includes a computer 600 and a computer peripheral, game control device 601. The computer is a conventional desktop PC or other type of computer. Computer 600 runs an operating system 602 such as the Microsoft "Windows" family of operating systems. The operating system provides various system services to one or more application programs 603 running on computer 600. Computer 600 also has a USB communications driver and port 604.

Control device 601 is one of any number of different types of controllers, such as a joystick, keypad, etc. It has a plurality of human-actuated controls such as buttons, sliders, and other well-known input and/or output controls. A physical example of such a controller is shown in Fig. 6.

The control device has a USB port 605 and communicates with computer 600 via a USB communications medium 606. In addition, control device 601 is configured to store various HID data 607 such as various device class descriptors defined by the HID standard. The HID descriptors enumerate and describe aspects of the control device's controls, and are formatted as specified by the HID standard. The control device has non-volatile memory 608 in which the HID data is stored. HID data can be requested and obtained by computer 600 through the USB communications cable.

structures, the controls are referenced by their HID string indexes. This allows subsequent correlation of the HID data with the non-HID data.

Prior to using a HID control device in this implementation, a computer application program on computer 600 retrieves both the HID descriptors 607 and the non-HID descriptor 610 from control device 601. Within the non-HID descriptor, the computer locates data corresponding to a particular genre, and extracts the appropriate control mappings from the non-HID descriptor. Each control mapping indicates a control (referenced by its HID string index) and an associated action. The computer then uses the specified string indexes to correlate the control mappings to the actual controls specified by the HID descriptor.

The described technique relies on a predefined definition of genres, as described in the previous section of this document. Within the non-HID data structures discussed below, the various genres and the actions defined within the genres are identified by different constants such as predefined numeric identifiers.

This is illustrated more clearly in Fig. 10, which show the structure and content of a non-HID genre descriptor 610 in accordance with the described embodiment. Descriptor 610 includes four major sections: a header section 611, a control section 612, a genre section 613, and a diagram section 614.

Header section 611 contains the following fields:

- *dwLength*—the total length of the genre descriptor;
- *bVersion*—the version number of the genre descriptor;
- *bNumControls*—the total number of controls present on the control device;
- *bNumGenres*—the total number of genre supported by the control device, in order of manufacturer's preference;

sets forth a different HID string index. This string index corresponds to the HID string index as found in the HID descriptors.

- *bOverlayNumber*—identifies the overlay associated with the control, with reference to the *bDiagramIndex* field of the diagram section (see below).
- *wXOffset*—the location in the X dimension of the control in the indicated overlay (*bOverlayNumber*) from the top left of the overlay.
- *wYOffset*—the location in the Y dimension of the control in the indicated overlay (*bOverlayNumber*) from the top left of the overlay.
- *wZOffset*—the location in the Z dimension of the control in the indicated overlay (*bOverlayNumber*) from the top left of the overlay. Used only in 3-dimensional diagrams.
- *wXNormal*—indicates information regarding the Normal in the X direction. The three normal coordinates (see immediately below) specify a 3-dimensional vector which is used to orient pointers or lead lines when showing different rotations of the device.
- *wYNormal*—indicates information regarding the Normal in the Y direction.
- *wZNormal*—indicates information regarding the Normal in the Z direction.
- *bNumPointerSegments*—the number of segments in the lead line corresponding to the control. The following *wXSegment, wYSegment, and wZSegment* fields are repeated for each indicated segment (up to a limit of four).
- *wXSegment*—X coordinate of the base of an arrow segment.

  wYSegment—Y coordinate of the base of an arrow segment.

- *bOverlayToDiagram*—if this diagram is an overlay, contains the *bDiagramIndex* of the underlying diagram. If this diagram is not an overlay, the *bOverlayToDiagram* field is set to zero.
- *bDiagramData*—the actual diagram data structure (.bmp, .jpg, .gif, etc.)

Fig. 11 shows methodological aspects of this embodiment. Methodological features include an act 670 of defining a plurality of application program genres. Each such genre includes potential actions to be performed in an application program of that genre.

Act 671 comprises running an application program that has been classified as a particular genre. The application program is responsive to a plurality of human-actuated controls on a peripheral control device such as control device 601.

An act 672 comprises querying the control device to obtain a non-HID genre descriptor of the type shown in Fig. 10. Generally, the genre descriptor indicates actions to be performed by an application program of a particular genre in response to respective controls of the control device. As discussed above, the genre descriptor references controls by there HID string indexes.

Act 673 comprises retrieving one or more HID descriptors from the control device to determine various details regarding the controls and also to determine their string indexes.

Act 674 comprises correlating the controls with actions as specified by the non-HID descriptor. Act 675 comprises responding to the controls by performing the correlated actions in the application program.

In the system shown in Fig. 11, these steps are performed in conjunction with system services that are independent of the application program, such as part

(device, interface, endpoint, or other). The primary types of requests specified in the "request type" field are the "standard" and "vendor" types, which will be discussed below.

- *bRequest*—a request code indicating one of a plurality of different commands to which the device is responsive.
- *wValue*—a field that varies according to the request specified by *bRequest*.
- *wIndex*—a field that varies according to request; typically used to pass an index or offset as part of the specified request.
- *wLength*—number of bytes to transfer if there is a subsequent data stage.

All USB devices are supposed to support and respond to "standard" requests—referred to herein as "USB-specific" requests. In a USB-specific request, the request type portion of the *bmRequestType* field contains a predefined value indicative of the "standard" request type.

Each different USB-specific request has a pre-assigned USB-specific request code, defined in the USB specification. This is the value used in the *bRequest* field of the device request, to differentiate between different USB-specific requests. For each USB-specific request code, the USB specification sets forth the meanings of *wValue* and *wIndex*, as well as the format of any returned data.

USB devices can optionally support "vendor" requests—referred to herein as "device-specific" requests. In a device-specific request, the request type portion of the *bmRequestType* field contains a predefined value indicate of the "vendor" request type.

In the case of device-specific requests, the USB specification does not assign request codes, define the meanings of *wValue* and *wIndex*, or define the format of

(meanings of *bIndex*, *bValue*, etc.) is as specified in the definition of the host-specific device request. Phase 677 is repeated as desired during subsequent operation, without repetition of initialization phase 100.

Fig. 13 shows more details regarding the initialization phase 676. Actions performed by the host are on the left, and actions performed by the device are on the right.

The host performs an action 680 of sending a GET_DESCRIPTOR device request to the peripheral device. The GET_DESCRIPTOR device request is a standard, USB-specific request, identified in a control transfer by the GET_DESCRIPTOR request code (*bRequest* = GET_DESCRIPTOR). The fields of the control transfer (discussed above in the background section) have values as follows:

- *bmRequestType* = 10000000 (binary), indicating a "device-to-host" transfer, a "standard" or "USB-specific" type request, and a device recipient.
- *bRequest* = GET_DESCRIPTOR. This is a constant (six) defined by the USB specification
- *wValue* = 03EE (hex). The high byte (03) indicates that the request is for a "string" descriptor, and the low byte is an index value that is predefined as a constant in the definition of the host-specified device request. In this example, it has been defined as EE (hex), but could be predefined as any other value.
- *wIndex* = 0.
- *wLength* = 12 (hex). This is the length of a host-specific request descriptor that will be returned in response to this request. In the described example, the length is 12 (hex).

example MSFT100 indicates that this descriptor is for an "MSFT" host-specific device request, version "100" or 1.00.

- *bVendorCode*—the device-specific request code that is to be associated with the host-specified device request.
- *bPad*—a pad field of one byte.

The host receives the host-specific request descriptor (block 684) and then performs an action 685 of checking or verifying the signature and version number found in the *qwSignature* field. The correct signature confirms that the device is configured to support host-specific request codes. If either the signature or version number are incorrect, the host assumes that the device does not support host-specific request codes, and no further attempts are made to use this feature.

The signature field of the host-specific request descriptor block is what provides backward compatibility. A non-compatible device (one that doesn't support host-specific request codes) might use the predetermined *wValue* 03EE (hex) to store some other type of descriptor, which will be returned to the host without any indication of problems. However, this will become apparent to the host after it examines the data in the location where the signature is supposed to be. If the signature is not found, the host knows that the returned descriptor is not of the type requested, and will assume that the device does not support host-specific request codes.

If the signature and version are confirmed in block 685, the host reads the device-specific request code from the *bVendorCode* field, and uses it in the future as a host-specific request code (block 686), to initiate the host-specific device request. When using the device, the host sends the host-specific device request by specifying the obtained device-specific request code as part of a control transfer. The device responds by performing one or more predefined actions or functions that

CLAIMS

1. A game control device that conforms to Universal Serial Bus (USB) device class definitions for Human Interface Devices (HIDs), comprising:

a plurality of human-actuated controls;

one or more HID descriptors that describe aspects of the human-actuated controls, the HID descriptors associating HID string indexes with the respective human-actuated controls;

control mappings corresponding to a plurality of application program genres, the control mappings indicating actions to be performed in application programs of particular genres in response to respective ones of the human-actuated controls, wherein the control mappings identify controls by their HID string indexes.

2. A game control device as recited in claim 1, the control mappings being indicated in data sets comprising:

a control section indicating the HID string indexes for the respective controls;

a genre section indicating actions to be performed in application programs of particular genres in response to respective ones of the human-actuated controls.

3. A computer peripheral comprising:

a plurality of human-actuated controls;

non-volatile memory containing control mappings corresponding to a plurality of application program genres, the control mappings indicating actions to be performed in application programs of particular genres in response to respective ones of the human-actuated controls.

8. A computer peripheral as recited in claim 0, the non-volatile memory containing a descriptor comprising:

a control section indicating string indexes for the respective controls, the string indexes corresponding to separately defined human device interface (HID) string indexes;

a genre section indicating the control mappings for the respective application program genres, the control mappings identifying controls by their HID string indexes.

9. A computer peripheral as recited in claim 0, the non-volatile memory containing a descriptor comprising:

a header section indicating the number of controls on the computer peripheral and the number of genres for which control mappings exist in the non-volatile memory;

a control section indicating string indexes for the respective controls;

a genre section indicating the control mappings for the respective application program genres;

a diagram section containing one more graphics images of the computer peripheral, the one or more graphics images identifying locations of the human-actuated controls on the computer peripheral.

10. A computer peripheral as recited in claim 0, the non-volatile memory also containing control data that indicates:

string indexes for the respective controls;

graphics overlays that identify the human-actuated controls on the computer peripheral;

14. A method comprising:

defining a plurality of application program genres;

running an application program that has been classified as a particular application program genre, wherein the application program is responsive to a plurality of human-actuated controls on a control device;

querying the control device to obtain a genre descriptor, the genre descriptor indicating actions to be performed by an application program of said particular application program genre in response to respective ones of the human-actuated controls.

15. A method as recited in claim 14, wherein the obtained genre descriptor comprises:

a control section indicating string indexes for the respective controls;

a genre section indicating the control mappings for the respective application program genres.

16. A method as recited in claim 14, further comprising:

retrieving one or more HID descriptors from the control device, the HID descriptors describing aspects of the human-actuated controls, the HID descriptors associating HID string indexes with the respective human-actuated controls;

wherein the obtained genre descriptor identifies the human-actuated controls by their HID string indexes.

20. A method as recited in claim 14, wherein the obtained genre descriptor comprises:

string indexes for the respective controls;

graphics overlays that identify the human-actuated controls on the control device;

coordinates of the graphics overlays;

coordinates for pointers to the human-actuated controls.

21. A method as recited in claim 14, wherein the obtained genre descriptor comprises:

a header section indicating the number of controls on the control device and the number of genres for which control mappings exist in the non-volatile memory;

a control section indicating string indexes for the respective controls, the control section also indicating graphics overlays that identify the human-actuated controls on the control device;

a genre section indicating the control mappings for the respective application program genres;

22. A method as recited in claim 14, wherein the obtained genre descriptor comprises one more graphics images that identify the locations of the human-actuated controls on the control device.

26. A computer-readable storage medium as recited in claim 23, wherein the obtained genre descriptor comprises:

a control section indicating string indexes for the respective controls, the string indexes corresponding to separately defined human device interface (HID) string indexes;

a genre section indicating the control mappings for the respective application program genres, the control mappings identifying controls by their HID string indexes.

27. A computer-readable storage medium as recited in claim 23, wherein the obtained genre descriptor comprises:

a header section indicating the number of controls on the control device and the number of genres for which control mappings exist in the genre descriptor;

a control section indicating string indexes for the respective controls;

a genre section indicating the control mappings for the respective application program genres;

a diagram section containing one more graphics images of the control device, the one or more graphics images identifying locations of the human-actuated controls on the control device.

28. A computer-readable storage medium as recited in claim 23, wherein the obtained genre descriptor comprises:

string indexes for the respective controls;

graphics overlays that identify the human-actuated controls on the control device;

32. A data transmission medium carrying a data structure comprising:

a header section indicating the number of human-actuated controls on a computer peripheral and the number of application program genres for which control mappings exist in the data structure;

a control section indicating HID string indexes for the respective controls on the computer peripheral;

a genre section indicating control mappings for the respective application program genres.

33. A data transmission medium as recited in claim 32, further comprising:

a diagram section containing one more graphics images of the computer peripheral, the one or more graphics images identifying locations of the human-actuated controls on the computer peripheral.

34. A data transmission medium as recited in claim 32, wherein the control section also indicates graphics overlays that identify the human-actuated controls on the computer peripheral.

35. A data transmission medium as recited in claim 32, further comprising a diagram section, the diagram section comprising graphics overlays that identify the human-actuated controls on the computer peripheral;

wherein the control section indicates coordinates of the graphics overlays and coordinates for pointers to the human-actuated controls.

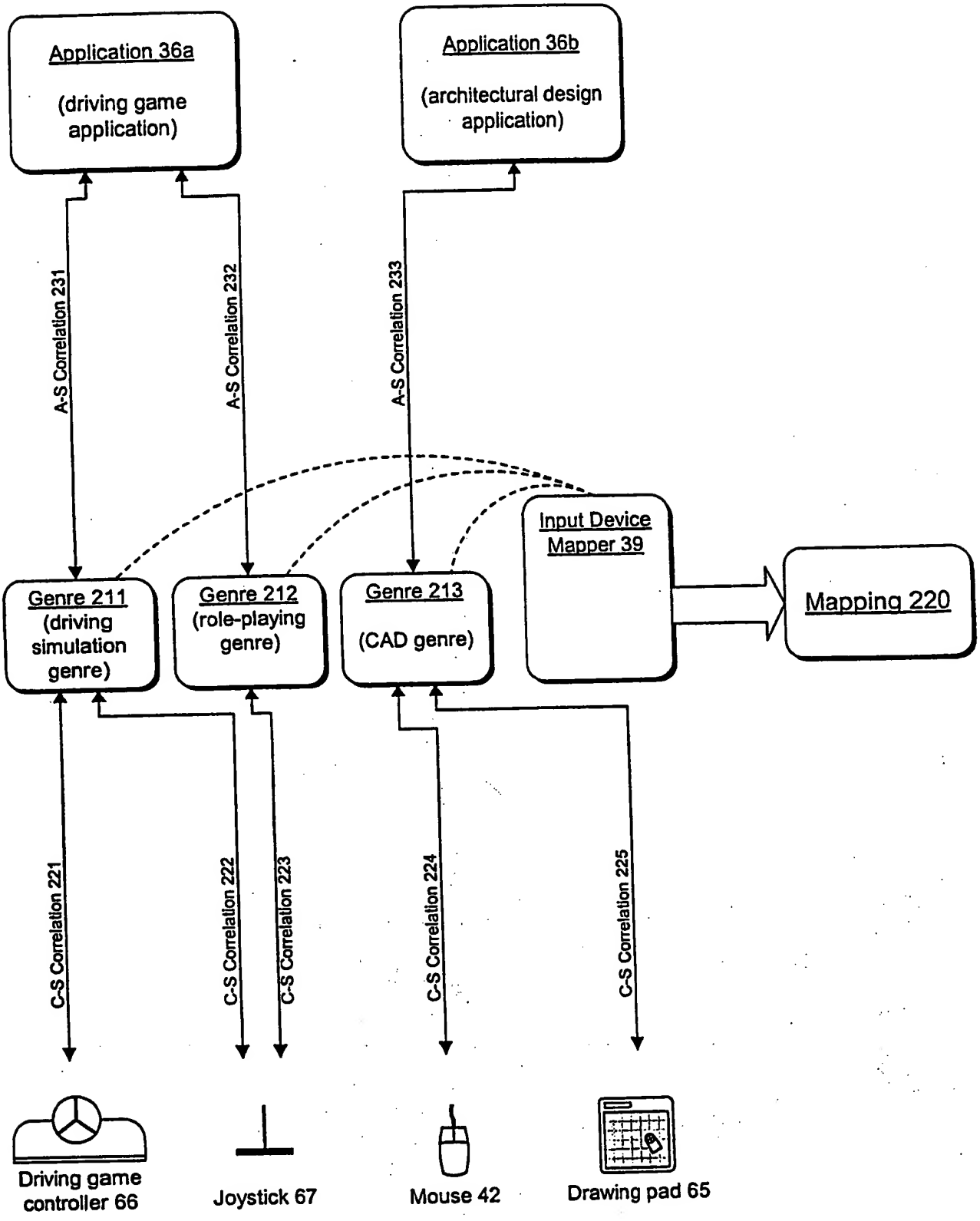

Application 36a
(driving game application)
Application 36b
(architectural design application)
A-S Correlation 231
A-S Correlation 232
A-S Correlation 233
Input Device Mapper 39
Mapping 220
Genre 211 (driving simulation genre)
Genre 212 (role-playing genre)
Genre 213 (CAD genre)
C-S Correlation 221
C-S Correlation 222
C-S Correlation 223
C-S Correlation 224
C-S Correlation 225
Driving game controller 66
Joystick 67
Mouse 42
Drawing pad 65

Figure 2

Figure 5

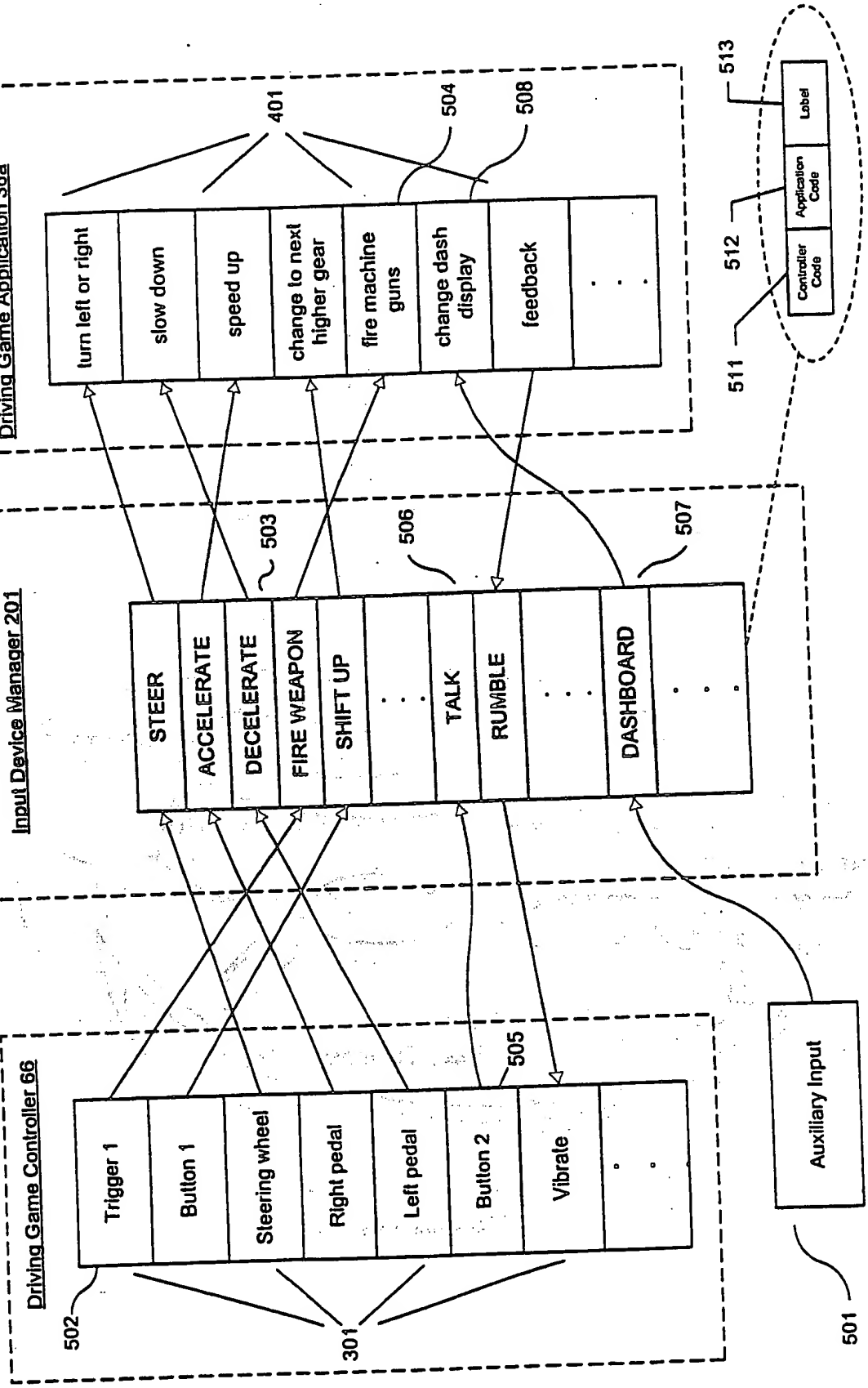
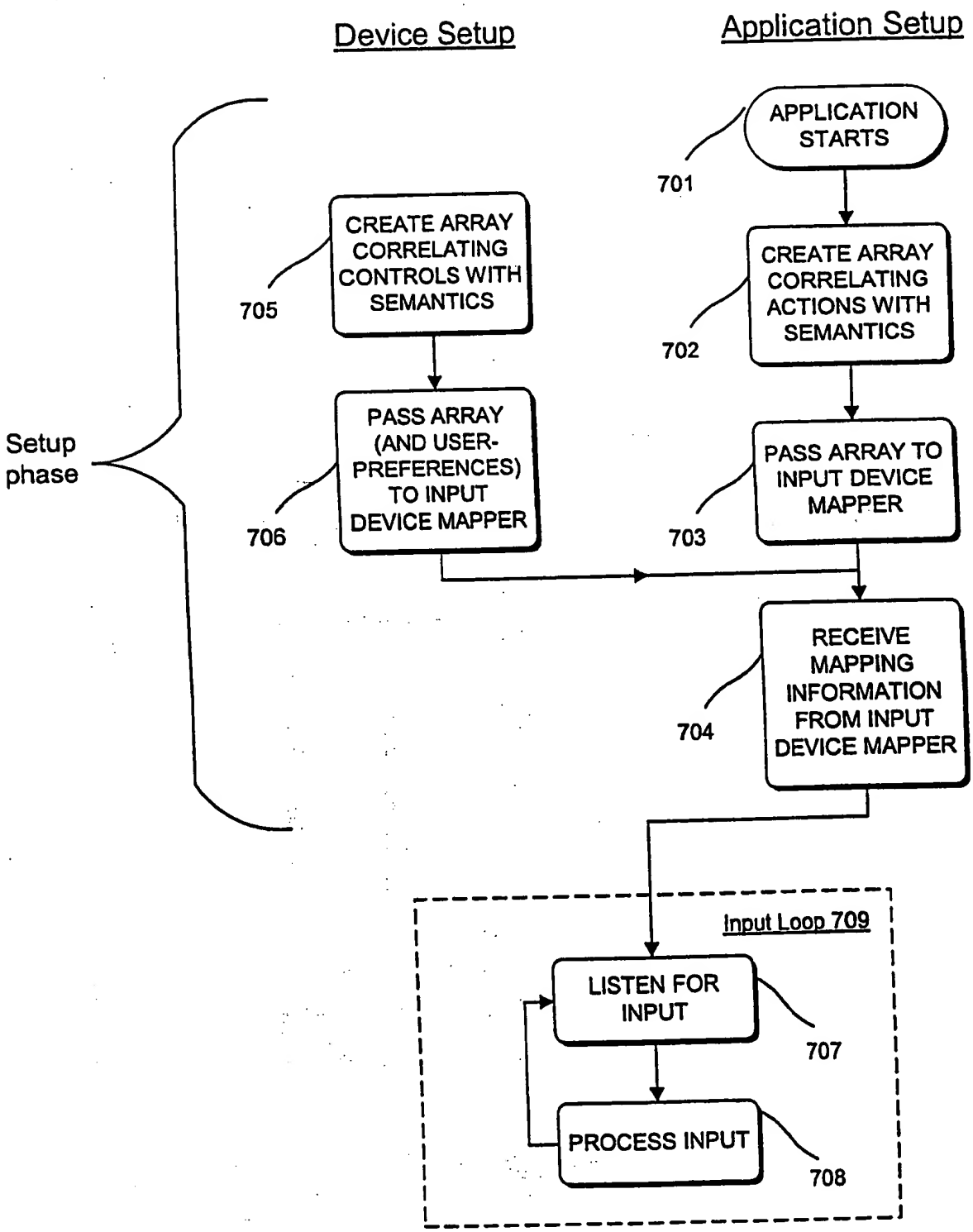

Mapping 220
Driving Game Application 36a
turn left or right
slow down
speed up
change to next higher gear
fire machine guns
change dash display
feedback
401
504
508
513
Label
Application Code
Controller Code
512
511
Input Device Manager 201
STEER
ACCELERATE
DECELERATE
FIRE WEAPON
SHIFT UP
TALK
RUMBLE
DASHBOARD
503
506
507
Driving Game Controller 66
Trigger 1
Button 1
Steering wheel
Right pedal
Left pedal
Button 2
Vibrate
505
502
301
Auxiliary Input
501

Figure 7


Device Setup
Application Setup
APPLICATION STARTS
701
CREATE ARRAY CORRELATING CONTROLS WITH SEMANTICS
705
CREATE ARRAY CORRELATING ACTIONS WITH SEMANTICS
702
Setup phase
PASS ARRAY (AND USER-PREFERENCES) TO INPUT DEVICE MAPPER
706
PASS ARRAY TO INPUT DEVICE MAPPER
703
RECEIVE MAPPING INFORMATION FROM INPUT DEVICE MAPPER
704
Input Loop 709
LISTEN FOR INPUT
707
PROCESS INPUT
708

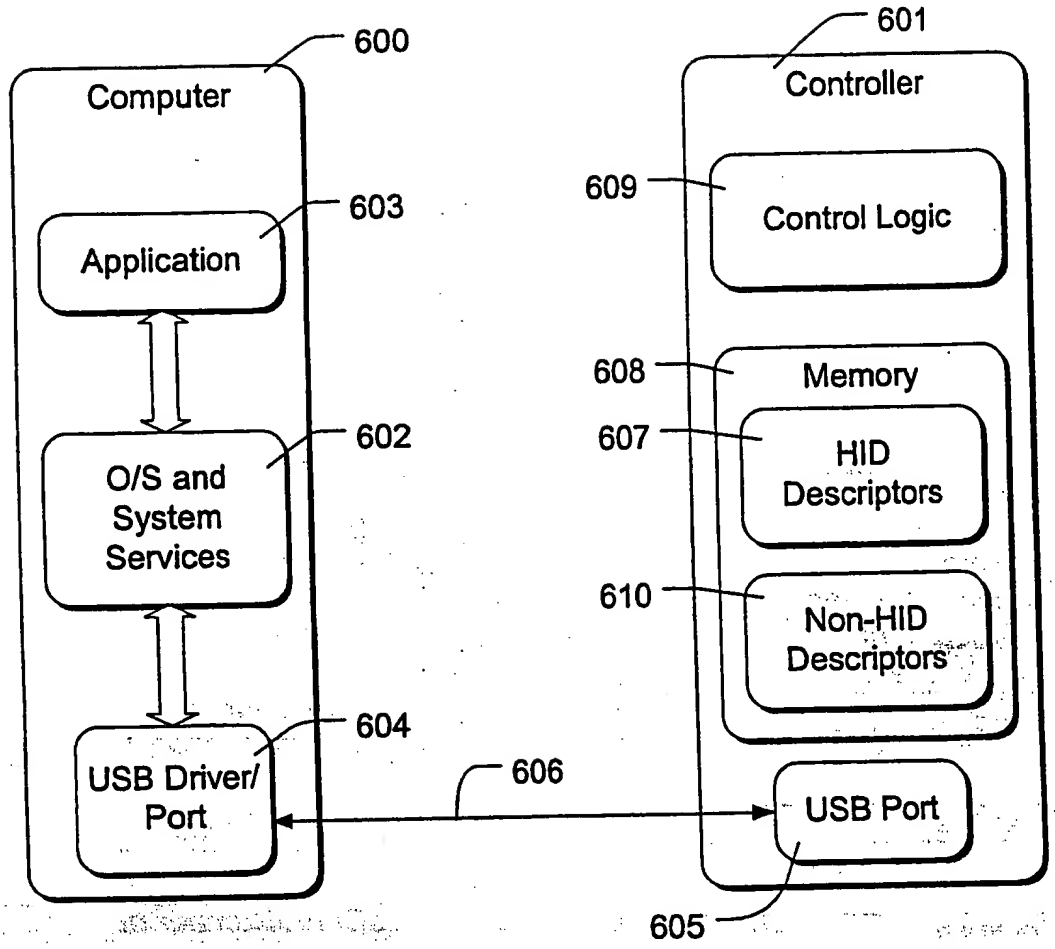

600
Computer
603
Application
602
O/S and System Services
604
USB Driver/ Port
606
601
Controller
609
Control Logic
608
Memory
607
HID Descriptors
610
Non-HID Descriptors
USB Port
605

Figure 9

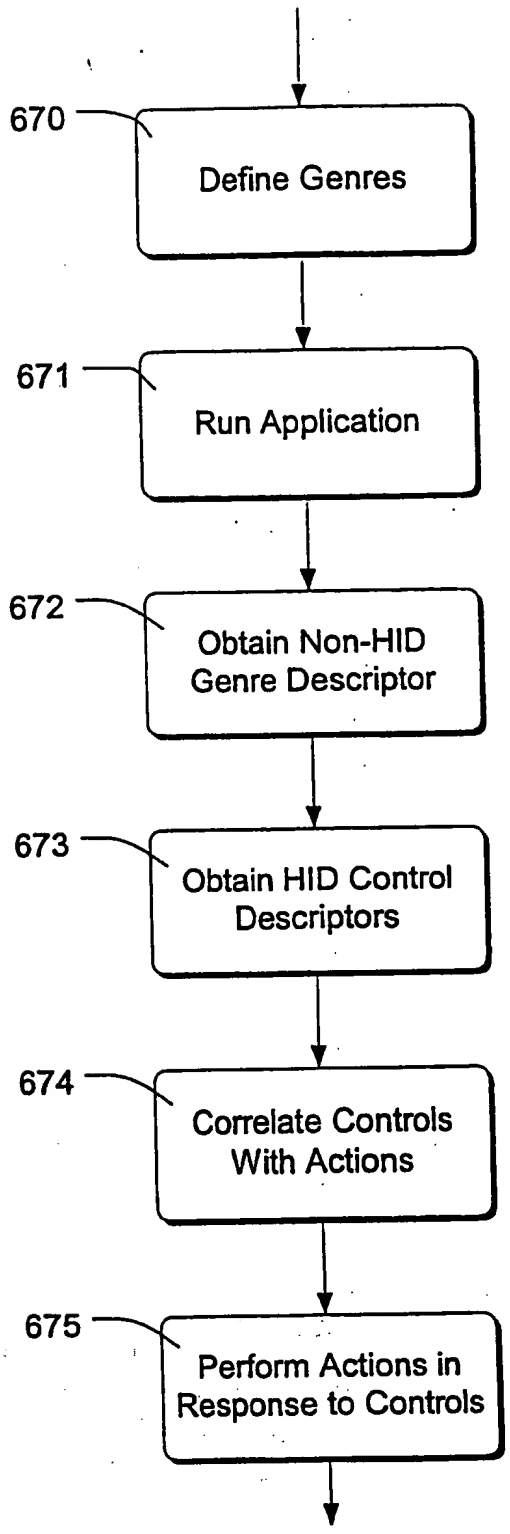

670
Define Genres
671
Run Application
672
Obtain Non-HID Genre Descriptor
673
Obtain HID Control Descriptors
674
Correlate Controls With Actions
675
Perform Actions in Response to Controls

Figure 11

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/08848

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
P,A	MIKE VAN FLANDERN (MICROSOFT) AND OTHERS: "Universal Serial Bus (USB) - Device Class Definition for Human Interface (HID)" 'Online! 4 July 1999 (1999-07-04) , USB IMPLEMENTERS' FORUM XP002143239 Retrieved from the Internet: <URL: http://www.usb.org/developers/data/devclass/hid1_1.pdf> 'retrieved on 2000-07-24! the whole document	1,3,14, 23,32

Figure 1

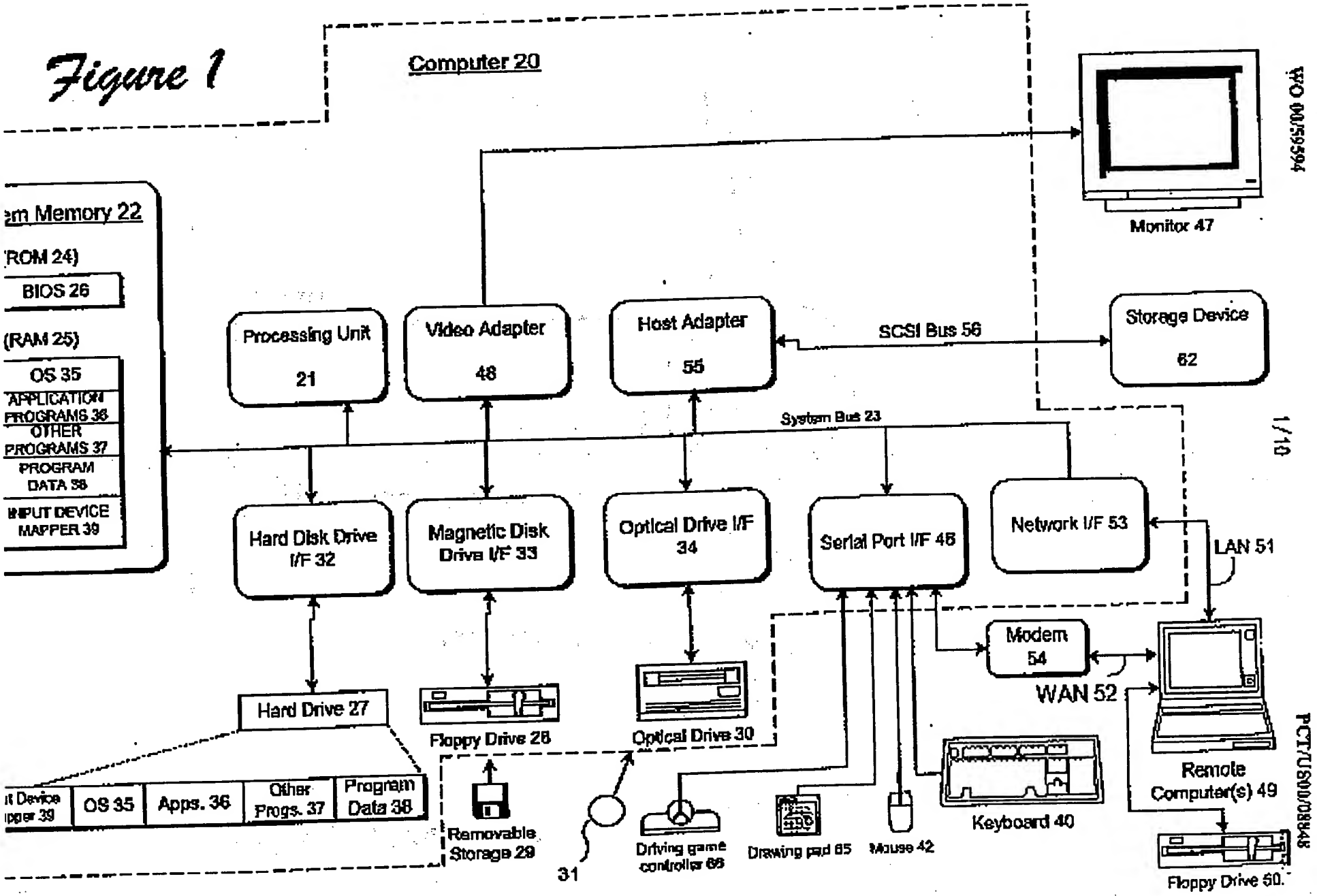


Figure 3

C-S Correlation 221

CONTROL ON DEVICE 66	SEMANTIC
Trigger 1	FIRE
Button 1	SHIFT UP
Steering Wheel	STEER
Right Pedal	ACCELERATE
Left Pedal	DECELERATE
Button 2	TALK
Vibrate	RUMBLE
.	.
.	.
.	.

301

302

Figure 4

A-S Correlation 231

ACTION IN APPLICATION 36a	SEMANTIC
turn left or right	STEER
speed up	ACCELERATE
slow down	DECELERATE
change to next higher gear	SHIFT UP
fire machine guns	FIRE
change dash display	DASHBOARD
feedback	RUMBLE
.	.
.	.
.	.

401

402

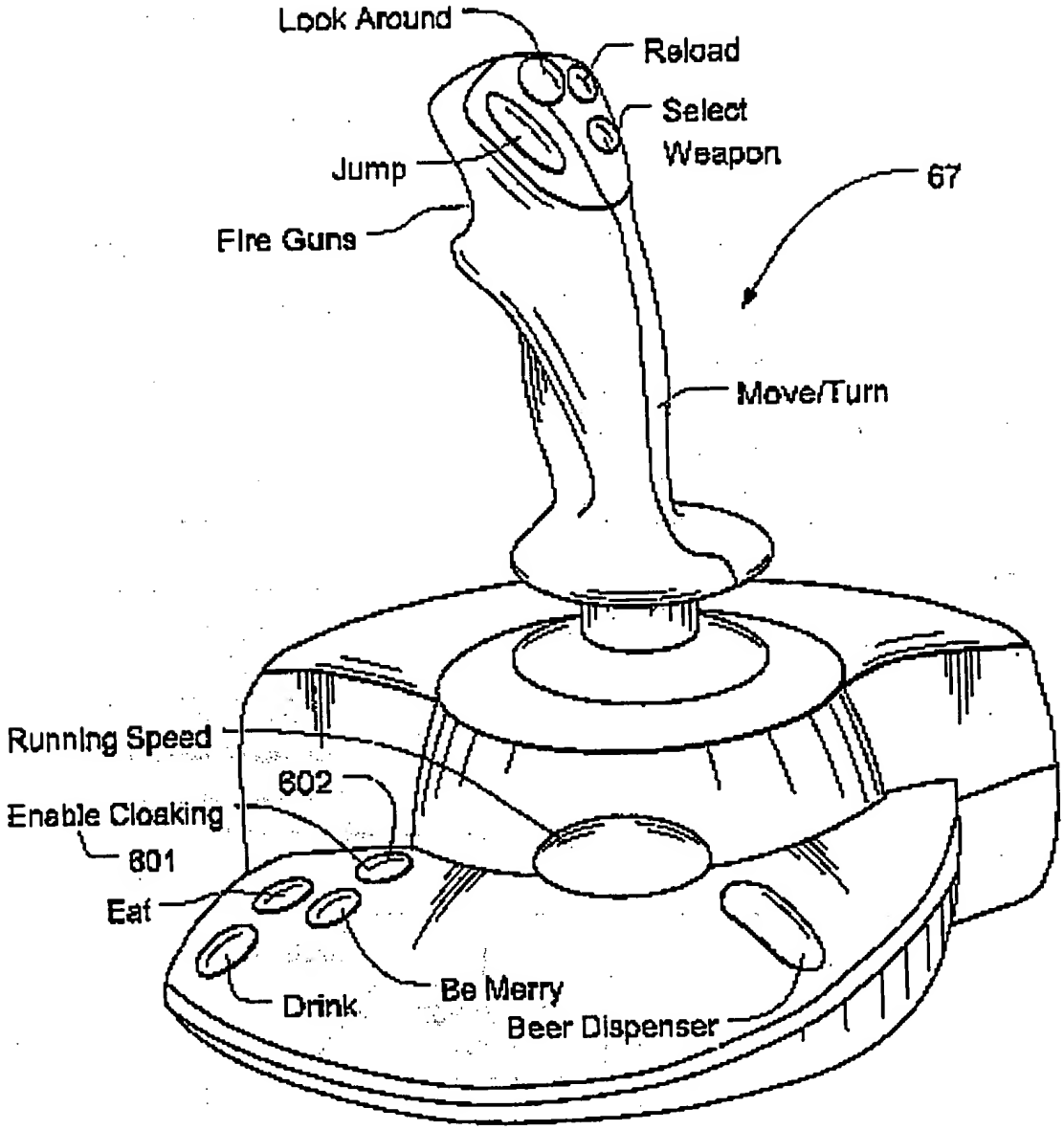

Look Around
Reload
Select Weapon
Jump
67
Fire Guns
Move/Turn
Running Speed
602
Enable Cloaking
601
Eat
Drink
Be Merry
Beer Dispenser

Figure 6

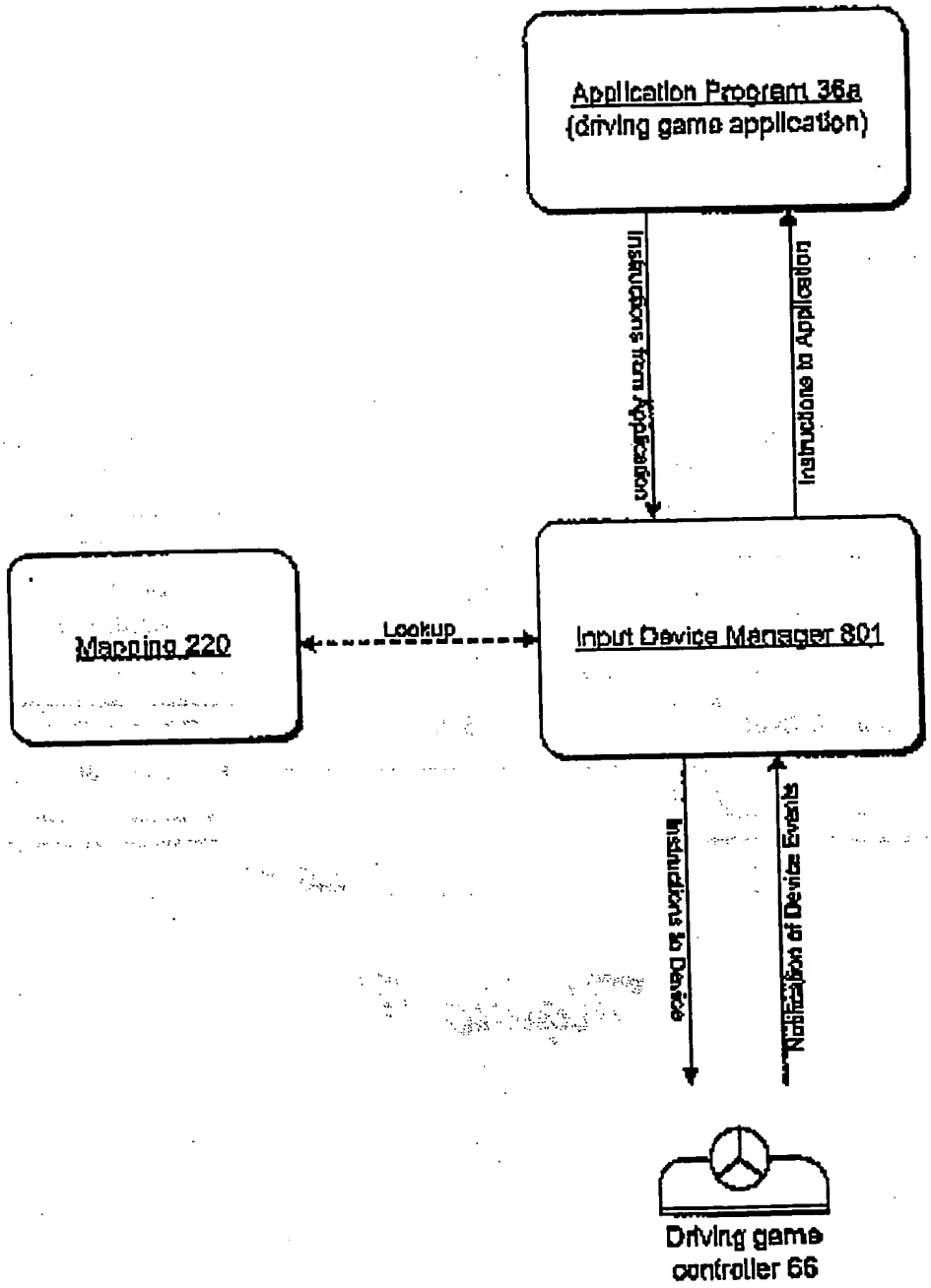

Application Program 36a
(driving game application)
Instructions from Application
Instructions to Application
Mapping 220
Lookup
Input Device Manager 801
Instructions to Device
Notification of Device Events
Driving game
controller 66

Figure 8

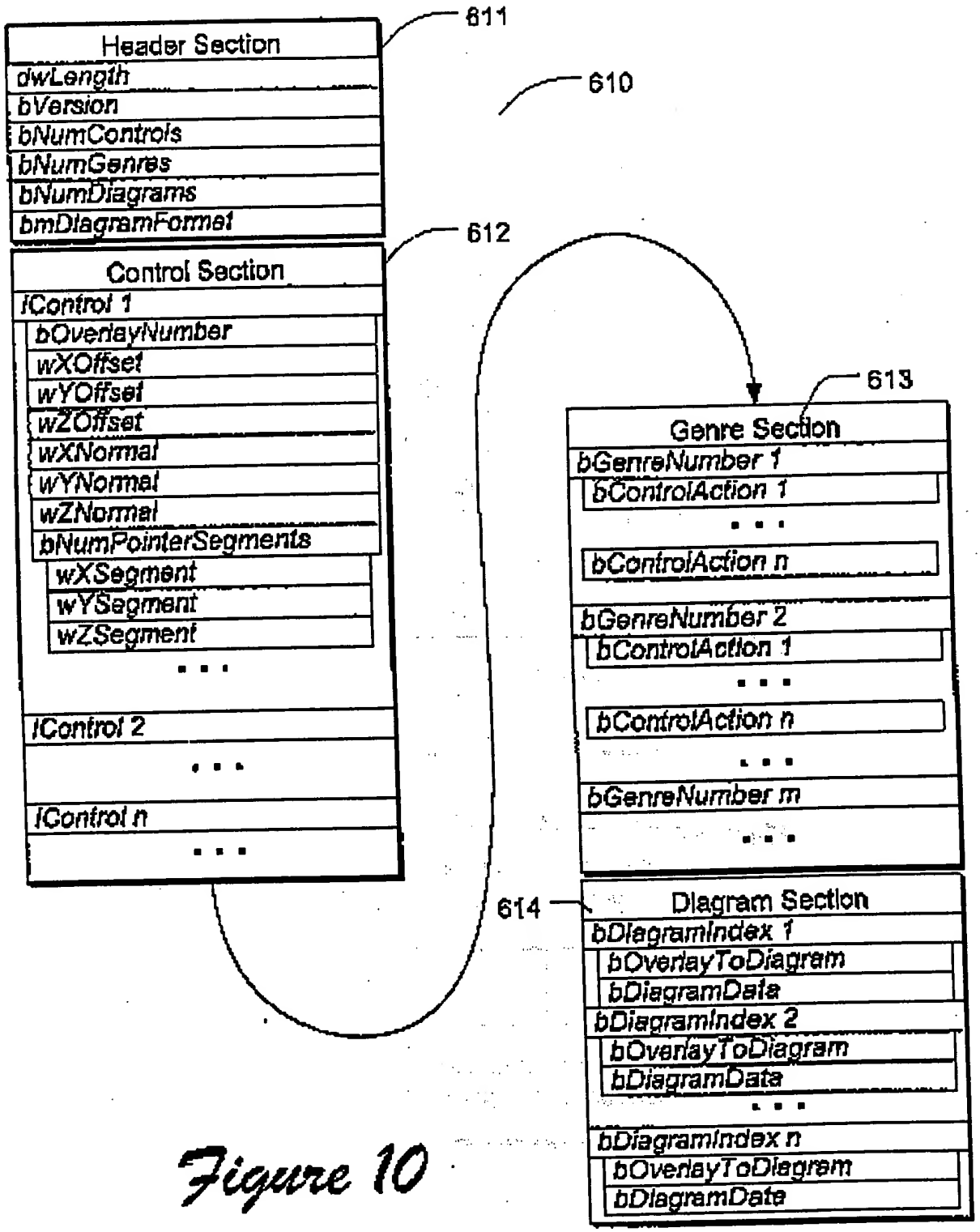

611
610
612
613
614
Header Section
dwLength
bVersion
bNumControls
bNumGenres
bNumDiagrams
bmDiagramFormat
Control Section
lControl 1
bOverlayNumber
wXOffset
wYOffset
wZOffset
wXNormal
wYNormal
wZNormal
bNumPointerSegments
wXSegment
wYSegment
wZSegment
lControl 2
lControl n
Genre Section
bGenreNumber 1
bControlAction 1
bControlAction n
bGenreNumber 2
bControlAction 1
bControlAction n
bGenreNumber m
Diagram Section
bDiagramIndex 1
bOverlayToDiagram
bDiagramData
bDiagramIndex 2
bOverlayToDiagram
bDiagramData
bDiagramIndex n
bOverlayToDiagram
bDiagramData

Figure 10